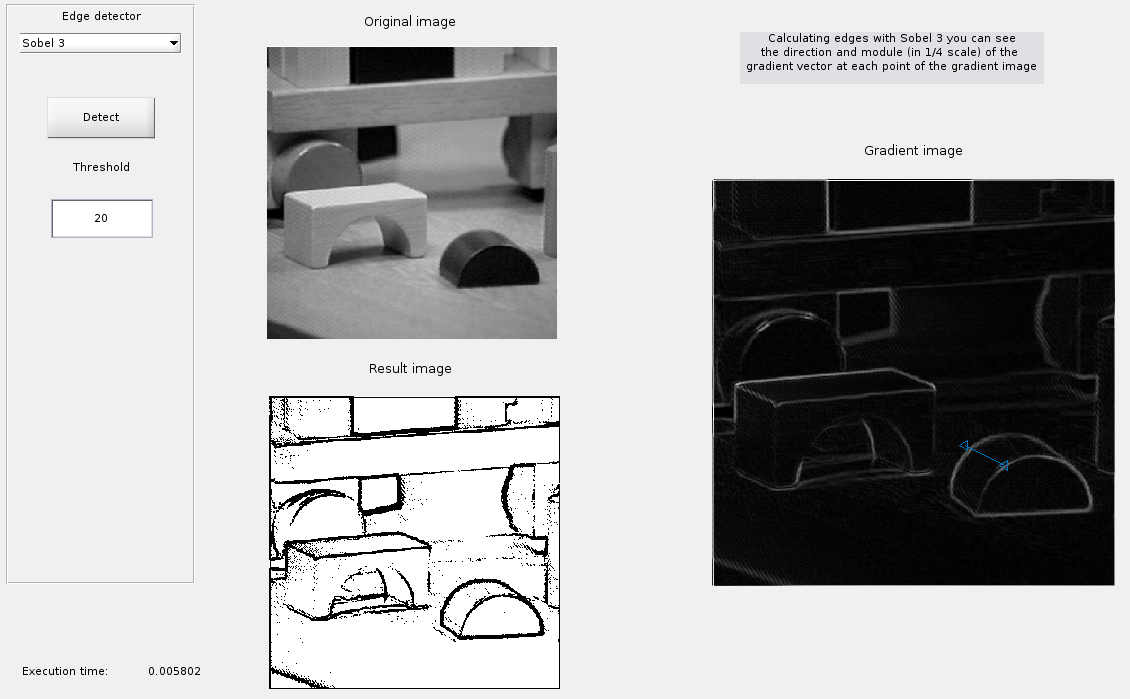
# COMPUTER VISION

**EXERCISE 2a: Edges Detection with mVision**

Concepts: Gradient operator: Sobel. Drog, Canny

1. Load the *edges* GUI from the *mVisionGUI* menu.
2. Load the *piezas.bmp* image.
3. Detect edges using Sobel3 (the Sobel operator with a 3x3 kernel) and Sobel5 (with a 5x5 kernel). You can play a bit with the gradient image in Sobel3 since it shows the gradient direction clicking on it.
   1. Which is the difference between them (in the result image) with the default threshold (20, pixel intensity in the gradient image for being considered as an edge)?
   2. Try both kernels with different thresholds: 20, 40 and 60. What is happening?
4. Now use the DroG detector with the default parameters.
   1. Why are detected less borders?
   2. Try with a different threshold and explain the result.
   3. With a threshold of 10, use different values for sigma (smoothing) and report which one provides the best output.
5. Finally, let’s try the Canny detector with the default values.
   1. Again, we are getting a poor edges image. What should we do?
   2. Matlab provides a function that detects edges in an image using different techniques, called edge(). Take a look at its syntax with help edge. Since through the GUI we are specifying just a threshold and a value for sigma, which are the values for the *large threshold* and the *small threshold* used by Canny?
   3. Modify the parameters for getting the best possible edges image. Would be that parameters valid for any image?
6. In your opinion, which is the best edge detector in terms of lower detection error, localization error, and multiple response?

****

**EXERCISE 2b: Implementing edges detection**

Concepts: Gradient operator: Sobel.

1. Design a Matlab function (not a script) that detects edges by means of the Sobel operator. The function must show a figure with 4 sub-plots (2x2), including:
   1. The initial image,
   2. an image with the vertical edges,
   3. an image with the horizontal edges,
   4. and a binary image with edges in green overlapping the initial one.

Input parameters of the function: image name, threshold for the edges image. It has to return a matrix with the edge angle in each pixel, in the range [-π…π].

**Useful functions**

|  |  |
| --- | --- |
| **img =conv2(img1, img2, type)** | Performs a convolution operation with **img1** and **img2**, taking into account the image border or not depending on the parameter **type**. |
| **mask = fspecial(mask\_type)** | Builds a mask of a certain type with 3x3 size (5x5 for LoG). |

**Result**

